Лабораторная работа 1-2

Проект заключается в разработке игры «Лабиринт сокровищ». В игре пользователь управляет персонажем в лабиринте, в котором находятся монстры и сокровища. Игроку нужно собрать как можно больше сокровищ за минимальное количество ходов и выйти из лабиринта.

Product backlog:

ID	Название	Важность (1-10)	Предварительная оценка	Как продемонстрировать	Примечания
1	Разработка главного меню программы	10	8	При запуске приложения появляется список с кнопками: • Новая игра • Загрузить игру • Статистика • Выход	Создание фрэймов с использованием Swing
2	Разработка поля ввода игрока	5	2	Чтобы начать игру нужно ввести имя пользователя в появившемся окне.	Валидация данных: - не должно содержать следующие слова: Sir, King, Queen, Lord, Lady - не должно заканчиваться римскими цифрами (I, IV и т.д.)
3	Разработка основного поля программы	8	16	После ввода имени игрока загружается игра. Вся игра проходит на карте (игровом поле), которая представляет из себя квадрат с ячейками внутри. Все объекты игры должны быть представлены графическими изображениями. Карта загружается из текстового файла с определенной структурой данных.	Создание фрэймов с использованием Swing. Создание нескольких игровых карт с разным расположением объектов на ней.

				Персонаж начинает игру с назначенной позиции и может двигаться по направлениям вверх, вниз, влево и вправо. Но не может по диагонали. Также игрок может пропустить ход, т.е. постоять на месте.	
4	Разработка стратегии движения объекта на поле	2	2	Смена стратегии меняет поведение объекта от быстрого движения по карте до медленного.	
5	Сохранение и восстановление игры	3	2	Игрок может в любое время выйти из игры, сохранив текущее положение всей карты с объектами. Следующий раз игрок может загрузить карту и продолжить игру. Если этот игрок проигрывает – сохраненная игра должна быть удалена.	
6	Вывод статистики по каждому игроку	3	4	При нажатии на поле «Статистика» на экран выводится таблица по всем игрокам со столбцами: • Имя игрока • Набранные очки • Общее количество игр • Количество удачных игр Таблица должна быть отсортирована по убыванию по столбцу «Набранные очки».	

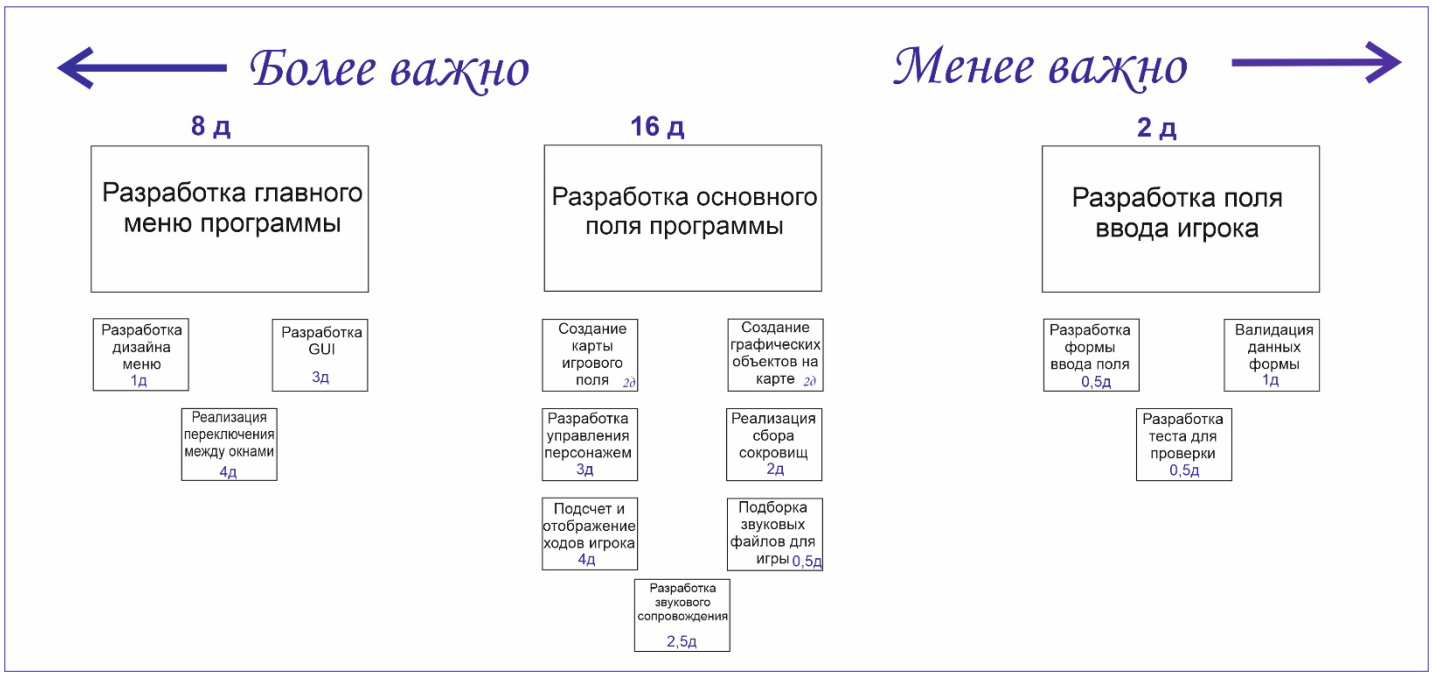
Спринт №1. Цель спринта – разработать графические элементы игры согласно техническому заданию и графический интерфейс игры.

Спринт №1 backlog

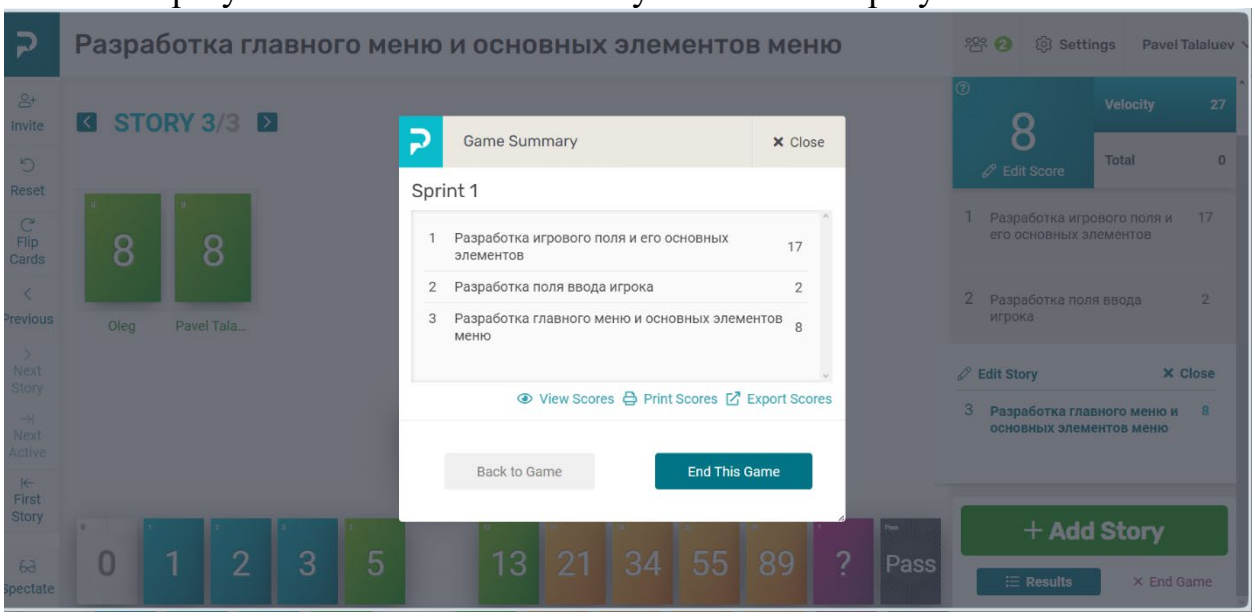
ID	Название	Производительность
1	Разработка главного меню программы	8
2	Разработка поля ввода игрока	2
3	Разработка основного поля программы	16

	Итого:	26

1. Разработка главного меню и основных элементов меню - 8
   1.1 Разработка дизайна меню – 1
   1.2 Разработка GUI - 3
   1.3 Реализация переключения между окнами - 4
2. Разработка поля ввода игрока - 2
   2.1. Разработка формы ввода поля – 0,5
   2.2. Валидация данных формы – 1
   2.3. Разработка теста для проверки – 0,5
3. Разработка игрового поля и его основных элементов - 16
   3.1.Создание карты игрового поля - 2
   3.2.Создание графических объектов на карте - 2
   3.3.Разработка управления персонажем, работа кнопок передвижения - 3
   3.4.Реализация сбора сокровищ - 2
   3.5.Подсчет и отображение ходов игрока - 4
   3.6.Подборка звуковых файлов для игры – 0,5
   3.7.Разработка звукового сопровождения ходов игрока – 2,5

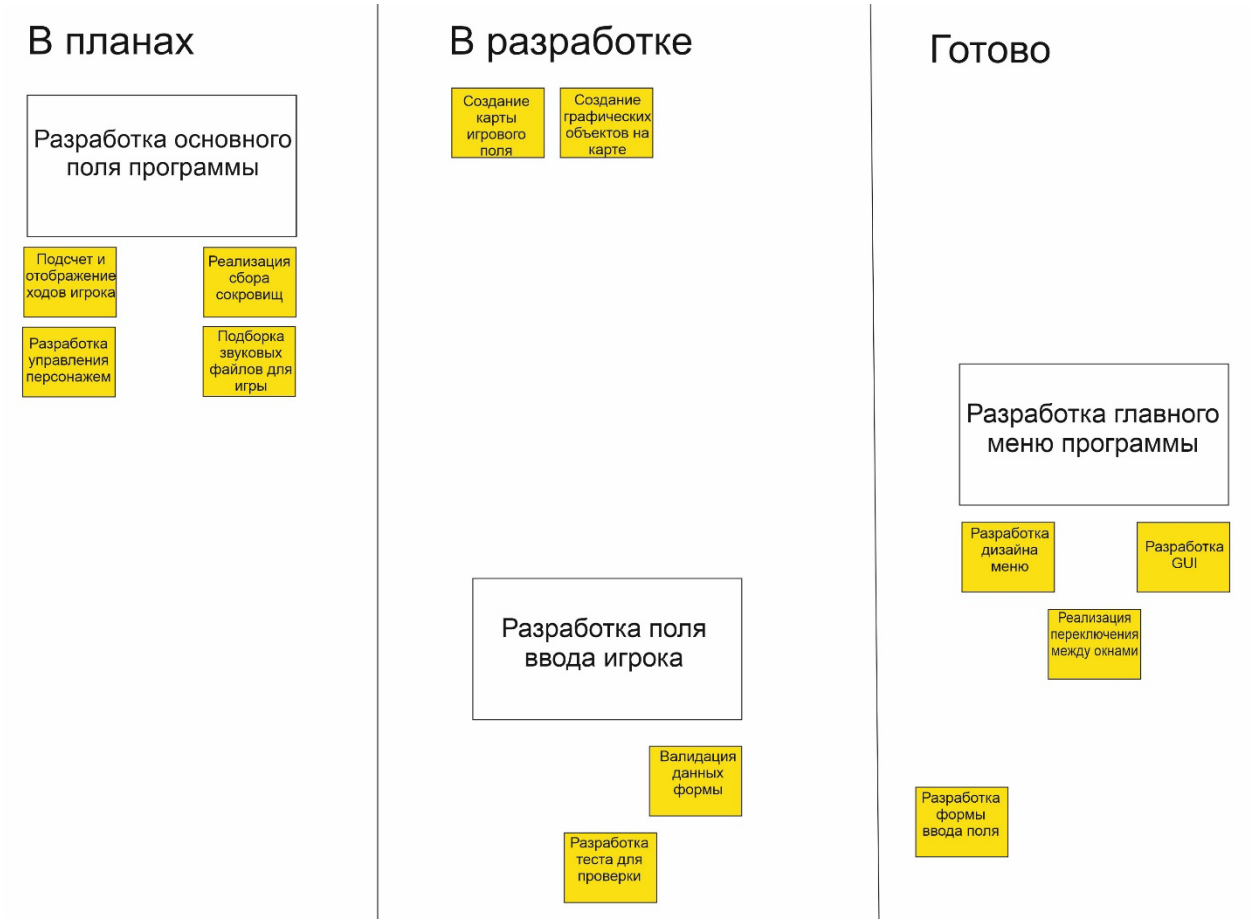


По результатам Scrum Poker получились такие результаты:



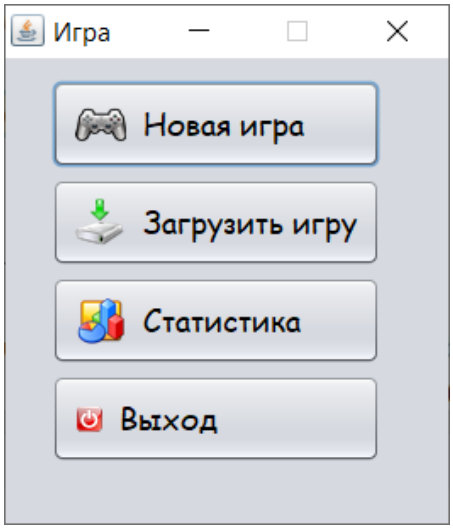
По результатам было принято решение на разработку игрового поля и его основных элементов выделить 17 дней, добавив задачу по тестированию полученных результатов и выделить на это 1 дополнительный день разработки.

Фактическое выполнение в таск трекере:

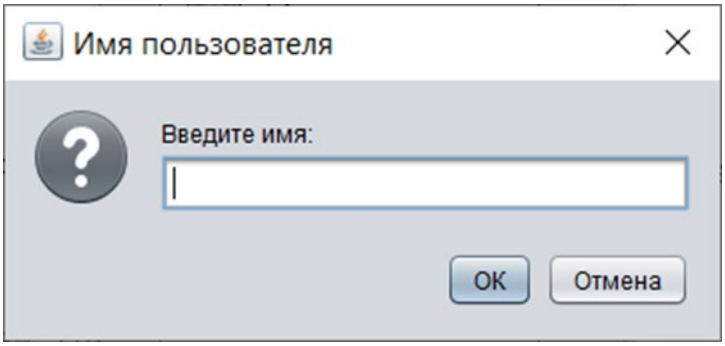


Результаты

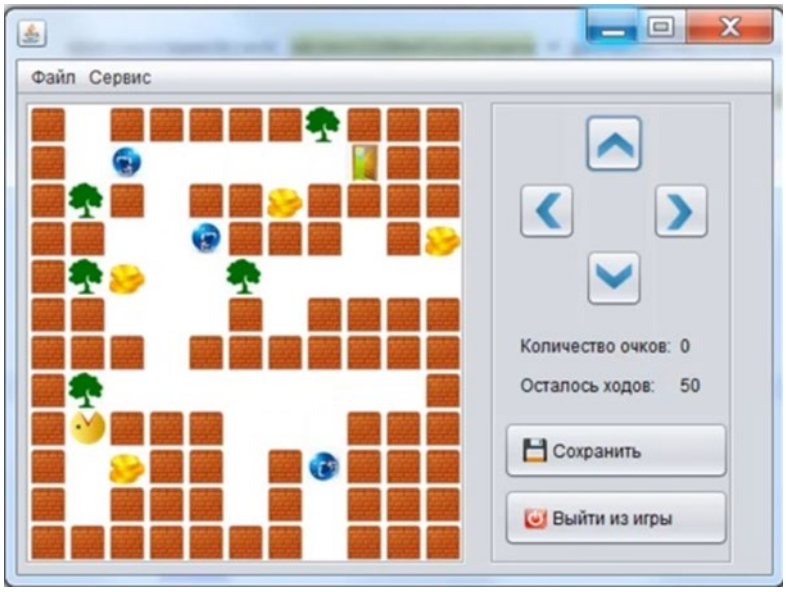
Главное меню программы:



Поле ввода игрока:



Основное поле программы:



Лабораторная работа 3

Часть 1. Проектирование архитектуры

1) Тип приложения – стандартная оконная программа под ОС Microsoft Windows, простое клиентское приложение.
2) Развертывание целесообразно использовать для больших систем. В простейшем случае программа не требует специальных действий для работы с ней. Однако для сложных систем для запуска комплекса может возникнуть необходимость в выполнении целого ряда работ. Например, может потребоваться развернуть аппаратное обеспечение или оборудование, инсталляция аппаратного и программного обеспечения для его работы, установка и настройка специального программного обеспечения: серверов приложений и БД, клиентских программ, заполнение баз начальными данными.
3) Основными факторами выбора технологии проектирования можно назвать следующие:
    - быстрота разработки приложения;
    - высокая производительность разработанного приложения;
    - низкие требования разработанного приложения к ресурсам компьютера;
    - возможность разработки новых компонент и инструментов собственными средствами;
    - наращиваемость (за счет встраивания новых компонент);
    - удобная проработка иерархии объектов.

    В соответствии с этими факторами были выбраны следующие технологии:
    - Среда разработки IntelliJ IDEA.
    - Язык программирования Java.
    - Библиотека Swing.

- Встраиваемая база данных SQLite.

4) Показатели качества

Функциональные требования

1. В приложении должно присутствовать меню для более удобного начала игры.

2. Пользователь должен иметь возможность включать и отключать музыкальное сопровождение во время игрового процесса.

3. Пользователь должен иметь возможность использования игровой способности.

Нефункциональные требования

1. Приложение должно быть написано на языке Java с использованием графического интерфейса swing.

2. Приложение должно при закрытии автоматически сохранять необходимые данные в реестр, а также автоматически загружать их при запуске приложения.

3. Системные требования:

- Операционная система: Windows 7, 10
- Процессор: 1.8 ГГц
- Оперативная память: 512 МБ
- Видеоадаптер: 3D, DirectX 7
- Свободного места на жестком диске: 20 МБ

Именно требования к приложению определяют показатели качества.

5) Поскольку приложение очень простое, в нем будет использована только валидация имени пользователя.

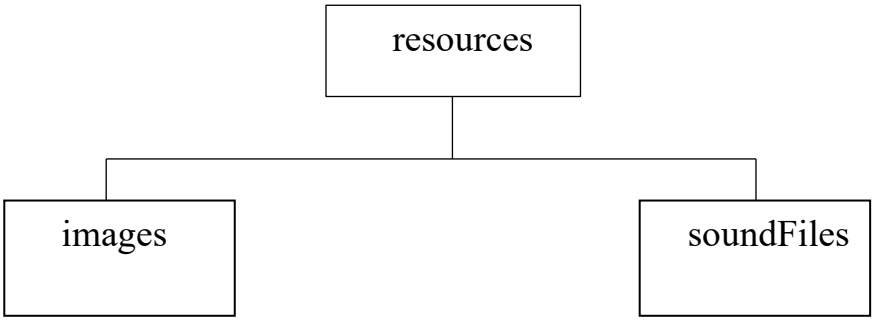
Требования к валидации имени пользователя:

- Не должно содержать следующие слова: Sir, King, Queen, Lord, Lady
- Не должно заканчиваться римскими цифрами (I, IV и т.д.)

Валидация будет применена к форме ввода имени игрока.

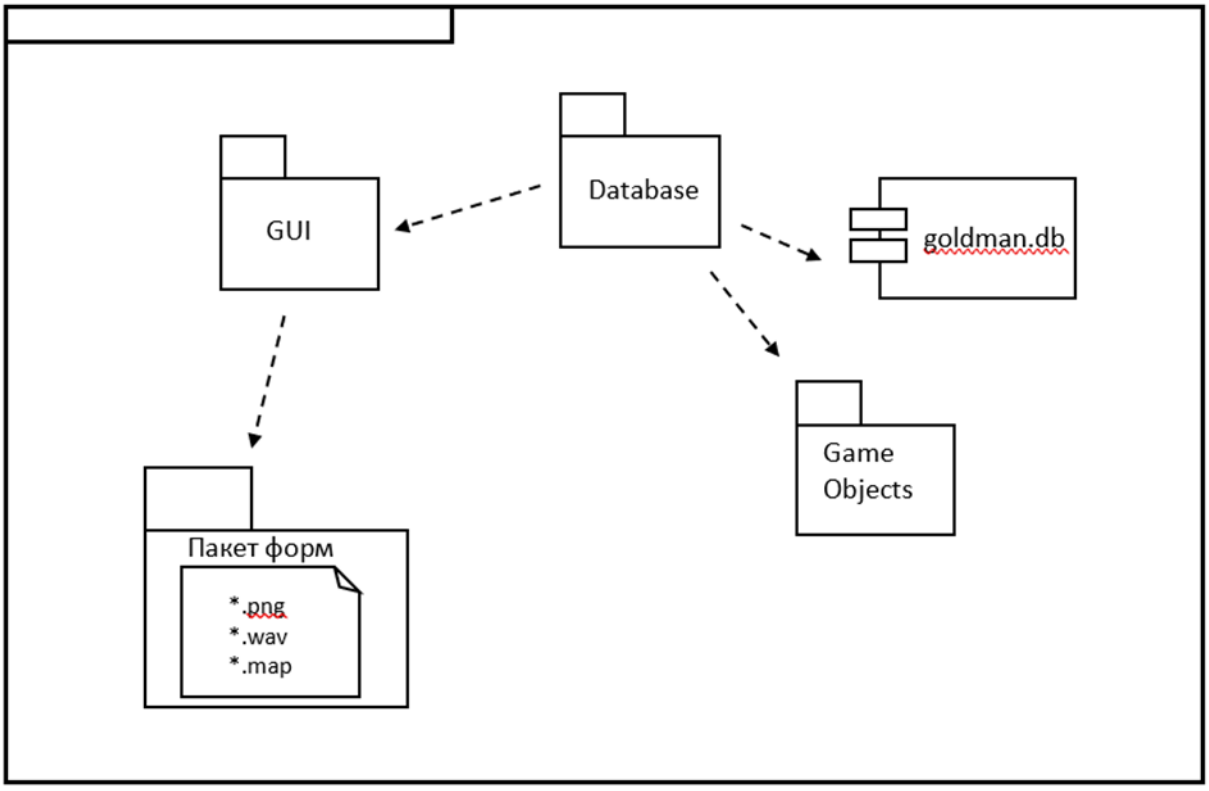
6. Диаграмм компонентов

Проект игрового приложения состоит из набора каталогов, содержащих файлы анимации и звукового сопровождения. Файловая структура приложения представлена на рис. ниже.



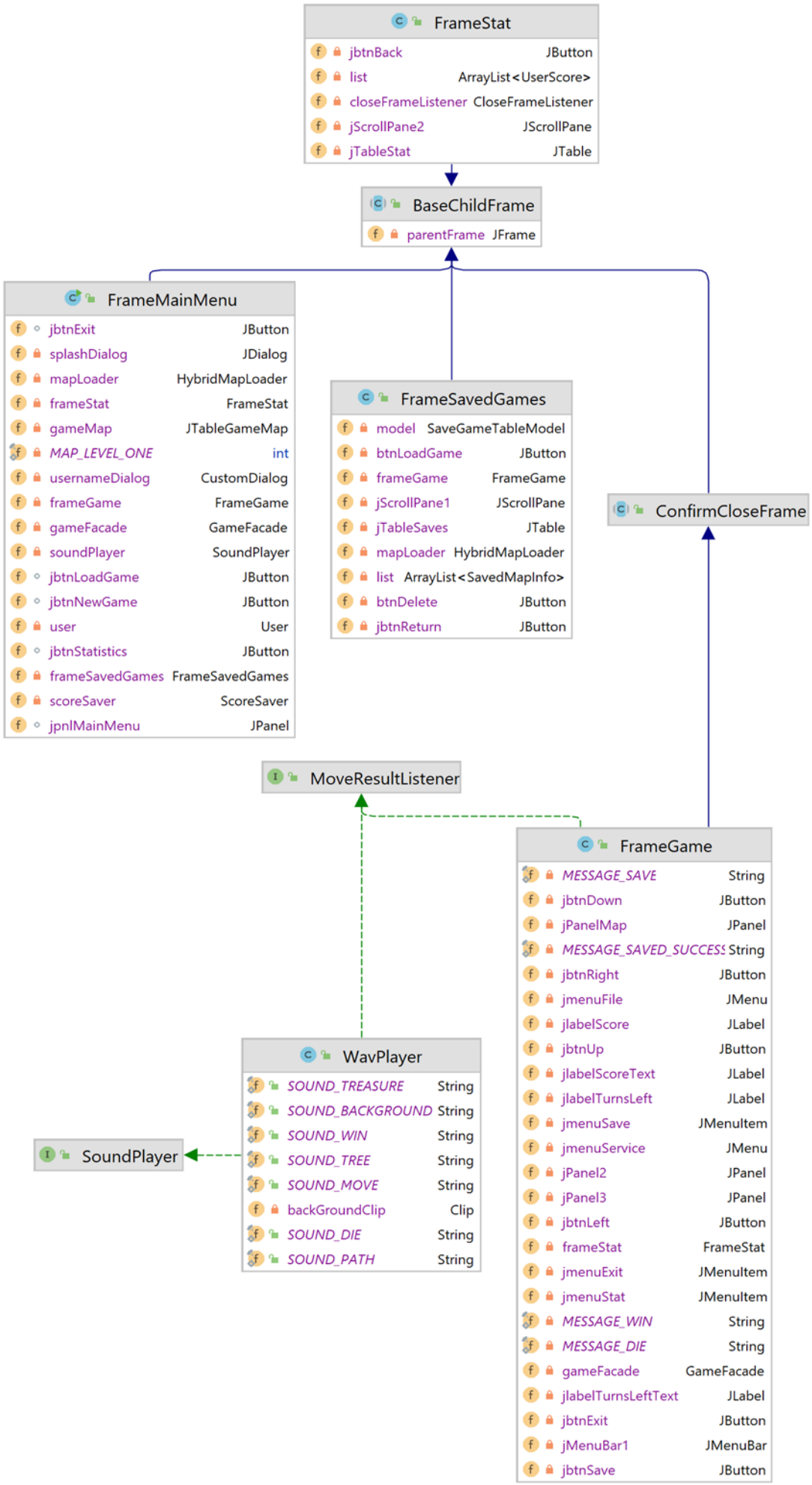
Пакет images содержит файлы для отображения игрового меню и поля. Пакет soundFiles содержит звуковые файлы.

Диаграмма компонентов представлена ниже.



Часть 2. Анализ архитектуры

Диаграммы классов (Class Diagrams) используется для моделирования статической структуры классов проектируемой системы и связей между ними. Ниже представлена диаграмма классов части компонентов GUI.


FrameStat
jbtnBack JButton
list ArrayList<UserScore>
closeFrameListener CloseFrameListener
jScrollPane2 JScrollPane
jTableStat JTable
BaseChildFrame
parentFrame JFrame
FrameMainMenu
jbtnExit JButton
splashDialog JDialog
mapLoader HybridMapLoader
frameStat FrameStat
gameMap JTableGameMap
MAP_LEVEL_ONE int
usernameDialog CustomDialog
frameGame FrameGame
gameFacade GameFacade
soundPlayer SoundPlayer
jbtnLoadGame JButton
jbtnNewGame JButton
user User
jbtnStatistics JButton
frameSavedGames FrameSavedGames
scoreSaver ScoreSaver
jpnlMainMenu JPanel
FrameSavedGames
model SaveGameTableModel
btnLoadGame JButton
frameGame FrameGame
jScrollPane1 JScrollPane
jTableSaves JTable
mapLoader HybridMapLoader
list ArrayList<SavedMapInfo>
btnDelete JButton
jbtnReturn JButton
ConfirmCloseFrame
MoveResultListener
FrameGame
MESSAGE_SAVE String
jbtnDown JButton
jPanelMap JPanel
MESSAGE_SAVED_SUCCESS String
jbtnRight JButton
jmenuFile JMenu
jlabelScore JLabel
jbtnUp JButton
jlabelScoreText JLabel
jlabelTurnsLeft JLabel
jmenuSave JMenuItem
jmenuService JMenu
jPanel2 JPanel
jPanel3 JPanel
jbtnLeft JButton
frameStat FrameStat
jmenuExit JMenuItem
jmenuStat JMenuItem
MESSAGE_WIN String
MESSAGE_DIE String
gameFacade GameFacade
jlabelTurnsLeftText JLabel
jbtnExit JButton
jMenuBar1 JMenuBar
jbtnSave JButton
WavPlayer
SOUND_TREASURE String
SOUND_BACKGROUND String
SOUND_WIN String
SOUND_TREE String
SOUND_MOVE String
backGroundClip Clip
SOUND_DIE String
SOUND_PATH String
SoundPlayer

Класс FrameMainMenu – запускает приложение и отображает графический интерфейс меню игры.

Класс BaseChildFrame – абстрактный класс, от которого наследуются классы создания дочерних фреймов статистики, фрейм сохраненного меню. Он используется для переключения окон в игре (родительское и дочернее окно).

Класс FrameStat – класс, отвечающий за разработку статистики по игре.

Класс FrameGame – класс, отвечающий за вывод сообщений на экран.

Класс FrameSavedGames – класс, отвечающий за сохранение игры.

Класс WavPlayer – класс, отвечающий за звуковое сопровождение игры.

Класс ConfirmCloseFrame – абстрактный класс, определяющий новое поведение для закрытия окна в классе FrameGame (переопределение метода с дополнительной функцией).

Часть 3. Сравнение и рефакторинг

Диаграмма классов в лабораторной работе отображает только один из компонентов, представленных в диаграмме компонентов (GUI), однако, стоит сказать, что структура классов проекта полностью следует диаграмме компонентов. Отличий нет, поскольку это довольно небольшой проект и в нем не так много компонентов для разработки.

Пути улучшения архитектуры – это использование шаблонов проектирования. Паттерны проектирования помогают наиболее эффективно решить задачи проектирования.

1) Паттерн проектирования «Singleton»

Используем единственный экземпляр класса для того, чтобы создать объекты в любом месте программы. Используется для подключения к базе данных.

```
private static GameObjectCreator instance;

public static GameObjectCreator getInstance() {
    if (instance == null) {
        instance = new GameObjectCreator();
    }
```

```
    return instance;
}


private static SQLiteConnection instance;

public static SQLiteConnection getInstance() {
    if (instance == null) {
        instance = new SQLiteConnection();
    }

    return instance;
}
```

2) Паттерн проектирования «Наблюдатель»

Наблюдатель слушает все ходы и делает все необходимые действия на карте и в коллекции.

```
public interface MoveResultNotifier {

    List<MoveResultListener> getMoveListeners();

    void addMoveListener(MoveResultListener listener);

    public void removeMoveListener(MoveResultListener listener);

    public void removeAllMoveListeners();

    public void notifyMoveListeners(ActionResult actionResult,
AbstractGameObject movingObject, AbstractGameObject targetObject);

}
```

3) Паттерн проектирования «Фасад»

Это структурный паттерн проектирования, который предоставляет простой интерфейс к сложной системе классов, библиотеке или фреймворку. Фасад — это простой интерфейс для работы со сложной подсистемой, содержащей множество классов. Фасад может иметь урезанный интерфейс, не имеющий 100% функциональности, которой можно достичь, используя сложную подсистему напрямую. Но он предоставляет именно те фичи, которые нужны клиенту, и скрывает все остальные.

```
    public class GameFacade {

    private HybridMapLoader mapLoader;
    private SoundPlayer soundPlayer;
    private ScoreSaver scoreSaver;
    private MapInfo mapInfo;
    private AbstractGameMap gameMap;

    public GameFacade(HybridMapLoader mapLoader, SoundPlayer soundPlayer,
ScoreSaver scoreSaver) {
```

```
        this.mapLoader = mapLoader;
        this.scoreSaver = scoreSaver;
        this.soundPlayer = soundPlayer;
    }

    public GameFacade() {
    }

    public void setSoundPlayer(SoundPlayer soundPlayer) {
        this.soundPlayer = soundPlayer;
    }

    public void setMapLoader(HybridMapLoader mapLoader) {
        this.mapLoader = mapLoader;

        // слушатели для звуков должны идти в первую очередь, т.к. они
запускаются в отдельном потоке и не мешают выполняться следующим слушателям
        if (soundPlayer instanceof MoveResultListener) {

mapLoader.getGameMap().getGameCollection().addMoveListener((MoveResultListene
r) soundPlayer); // реализация Паттерна Наблюдатель
        }

        updateMap();
    }

    public ScoreSaver getScoreSaver() {
        return scoreSaver;
    }

    public void setScoreSaver(ScoreSaver scoreSaver) {
        this.scoreSaver = scoreSaver;
    }

    public void stopGame() {
        soundPlayer.stopBackgoundMusic();
        mapLoader.getGameMap().stop();
    }

    public void moveObject(MovingDirection movingDirection, GameObjectType
gameObjectType) {
        gameMap.getGameCollection().moveObject(movingDirection,
gameObjectType);
    }

    public Component getMap() {
        return mapLoader.getGameMap().getMapComponent();
    }

    public void saveScore() {
        UserScore userScore = new UserScore();
        userScore.setUser(mapInfo.getUser());
        userScore.setScore(getGoldMan().getTotalScore());
        scoreSaver.saveScore(userScore);
    }

    public void addMoveListener(MoveResultListener listener) {
        mapLoader.getGameMap().getGameCollection().addMoveListener(listener);
// реализация Паттерна Наблюдатель
    }

    public void saveMap() {
        SavedMapInfo saveMapInfo = new SavedMapInfo();
        saveMapInfo.setId(mapInfo.getId());
```

```
        saveMapInfo.setUser(mapInfo.getUser());
        saveMapInfo.setTotalScore(getGoldMan().getTotalScore());
        saveMapInfo.setTurnsCount(getGoldMan().getTurnsNumber());
        mapLoader.saveMap(saveMapInfo, LocationType.DB);
    }

    public void startGame() {
        soundPlayer.startBackgroundMusic(WavPlayer.SOUND_BACKGROUND);
        mapLoader.getGameMap().start();
    }

    private GoldMan getGoldMan() {
        return (GoldMan)
mapLoader.getGameMap().getGameCollection().getGameObjects(GameObjectType.GOLD
MAN).get(0);
    }

    public int getTurnsLeftCount() {
        return mapInfo.getTurnsLimit() - getGoldMan().getTurnsNumber();
    }

    public int getTotalScore() {
        return getGoldMan().getTotalScore();
    }

    public void updateMap() {
        gameMap = mapLoader.getGameMap();
        mapInfo = gameMap.getMapInfo();


        gameMap.updateMap();
    }

    public void updateObjects(AbstractGameObject obj1, AbstractGameObject
obj2) {
        gameMap.updateMapObjects(obj1, obj2);
    }
}
```

4) Паттерн проектирования «Адаптер».

Адаптер — это структурный паттерн проектирования, который позволяет объектам с несовместимыми интерфейсами работать вместе.

```
public class HybridMapLoader {

    private DBMapLoader dBMapLoader;
    private FSMapLoader fSMapLoader;

    private AbstractGameMap gameMap;

    public HybridMapLoader(AbstractGameMap gameMap) {
        dBMapLoader = new DBMapLoader(gameMap);
        fSMapLoader = new FSMapLoader(gameMap);
        this.gameMap = gameMap;
    }

    public boolean saveMap(SavedMapInfo mapInfo, LocationType locationType){
        switch (locationType){
            case DB:{
                return dBMapLoader.saveMap(mapInfo);
            }
```

```
            case FS:{
                return fSMapLoader.saveMap(mapInfo);
            }
        }

        return false;
    }

    public boolean loadMap(MapInfo mapInfo, LocationType locationType){
        switch (locationType){
            case DB:{
                gameMap = dBMapLoader.getGameMap();
                return dBMapLoader.loadMap(mapInfo);
            }

            case FS:{
                gameMap = fSMapLoader.getGameMap();
                return fSMapLoader.loadMap(mapInfo);
            }
        }


        return false;
    }

    public ArrayList<SavedMapInfo> getSavedMapList(User user, LocationType
locationType){
        switch (locationType){
            case DB:{
                return dBMapLoader.getSavedMapList(user);
            }

            case FS:{
                return fSMapLoader.getSavedMapList(user);
            }
        }

        return null;
    }

    public boolean deleteSavedMap(MapInfo mapInfo, LocationType
locationType){
        switch (locationType){
            case DB:{
                return dBMapLoader.deleteSavedMap(mapInfo);
            }

            case FS:{
                return fSMapLoader.deleteSavedMap(mapInfo);
            }
        }

        return false;
    }

    public AbstractGameMap getGameMap() {
        return gameMap;
    }


    public int getPlayerId(String username){
        return dBMapLoader.getPlayerId(username);
    }
}
```

Лабораторная работа 4

Написать свои впечатления о парном программировании:

1. *Какую задачу реализовывали?*

Совместно разрабатывали приложение-игру на языке программирования Java, это была совместная лабораторная работа. Нужно было разработать простую игру, реализовать паттерны программирования.

Описание приложения.

В игре пользователь управляет персонажем в лабиринте, в котором находятся монстры и сокровища. Игроку нужно собрать как можно больше сокровищ за минимальное количество ходов и выйти из лабиринта.

За каждое собранное сокровище игрок получает очки. Если игрок будет съеден монстром – игра заканчивается с набранными очками. Также, если игрок не успевает выйти из лабиринта за определенное число шагов – игра заканчивается.

Игра содержит статистику по всем пользователям и лучшими результатами.

2. *Какой стиль показался более приемлемым? Почему?*

Наиболее приемлемым был стиль ведущий-ведомый, поскольку один человек в команде хорошо владел языком программирования Java, а другой только изучал Java и не имел практического опыта, поэтому второму человеку нужна была помощь в применении новых знаний. Такая работа показалась более эффективной, один писал код, второй вносил правки, корректировал, делал рефакторинг и помогал в разработке стратегий, подсказывал какую библиотеку лучше использовать, какой паттерн лучше применить.

3. *Что получилось? Положительные впечатления.*

Даже самый опытный разработчик не может знать всего. Поэтому мне кажется, что задачи решались быстрее и когда возникали сложные моменты было легче их решить.

4. Что не получилось? Как можно это поправить?

Психологически удобнее работать одному, кроме того, код каждого разработчика индивидуален в той или иной степени. Всегда есть человеческий фактор. Для парного программирования необходимо иметь терпение и желание работать в паре. Если один из участников будет просто сидеть молча и смотреть, как второй пишет код, или если один из программистов станет продавливать свое мнение, не прислушиваясь к коллеге, то такой вариант разработки не принесет ни удовольствия, ни эффективности.

5. Будете ли использовать ПП для других задач? Каких?

Часто парное программирование используется в менторстве, думаю, на работе могу столкнуться с такой задачей, когда нужна будет помощь в работе над чем-то новым, как этап обучения. Скорее всего работодателю будет неэффективно парное программирование, поскольку решение одной задачи двумя работниками – это более высокие трудовые затраты.

Лабораторная работа 5

Инспектирование технического долга по критериям:

№	Признаки технического долга	Комментарии
1	непонятный / нечитабельный код	По данному пункту замечаний не выявлено, названия методов и классов соответствуют сути. Используется единое стилевое решение, а также код соответствует принципам ООП, созданы абстрактные классы, интерфейсы, которые помогают код делать более читабельным и функциональным.
2	дублирующийся код	Дублирующийся код встречается в двух классах, повторяются методы.
3	отсутствие автоматизации (тестов, сборки, развёртывания)	Не весь код покрыт тестами
4	запутанная архитектура и ненужные сложные зависимости	Необходимо добавить стратегию по разбивке считывания данных не только с БД, но и с файловой системы.
5	медленные / неэффективные средства	Не выявлено
6	незакоммиченый код / долгоживущие ветки	Весь код коммитится по мере разработки и выполнения заданий
7	отсутствие / несоответствие технической документации	В техническую документацию необходимо внести пункт по покрытию кода тестами и разработки стратегии считывания данных с файла.
8	отсутствие тестовой среды	Не выявлено
9	длинные циклы интеграции / отсутствие непрерывной интеграции	Не выявлено

План мероприятий:

№	Признаки технического долга	Мероприятия
1	дублирующийся код	Следует использовать извлечение суперкласса, чтобы создать для интересующих классов один суперкласс, содержащий всю общую функциональность.

2	отсутствие автоматизации (тестов, сборки, развёртывания)	Необходимо разработать дополнительные тесты
3	запутанная архитектура и ненужные сложные зависимости	Необходимо добавить новую фичу по считыванию данных из файла
4	отсутствие / несоответствие технической документации	Необходимо дополнить техническую документацию и внести изменения

Оценка планируемых мероприятий:

№	Признаки технического долга	Производительность (дни)
1	дублирующийся код	2
2	отсутствие автоматизации (тестов, сборки, развёртывания)	2
3	запутанная архитектура и ненужные сложные зависимости	3
4	отсутствие / несоответствие технической документации	1

Необходимость устранения технического долга обусловлена с внесением новой функции, поскольку изначально она не была предусмотрена, также необходимостью создания тестов для избежания ошибок, дополнения технической документации и устранения узких мест в коде, которые возникли из-за спешки в разработке и отсутствия видения полной картины. По результатам оценки технического долга, срок разработки увеличится на 8 дней в связи с необходимостью его устранения и добавление новой функции.

ID	Название	Производительность
1	Разработка главного меню программы	8
2	Разработка поля ввода игрока	2
3	Разработка основного поля программы	16
4	Устранение технического долга	8
	Итого:	**34**

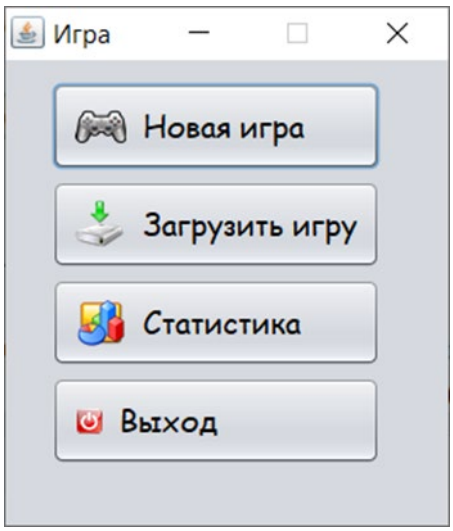
Лабораторная работа 6

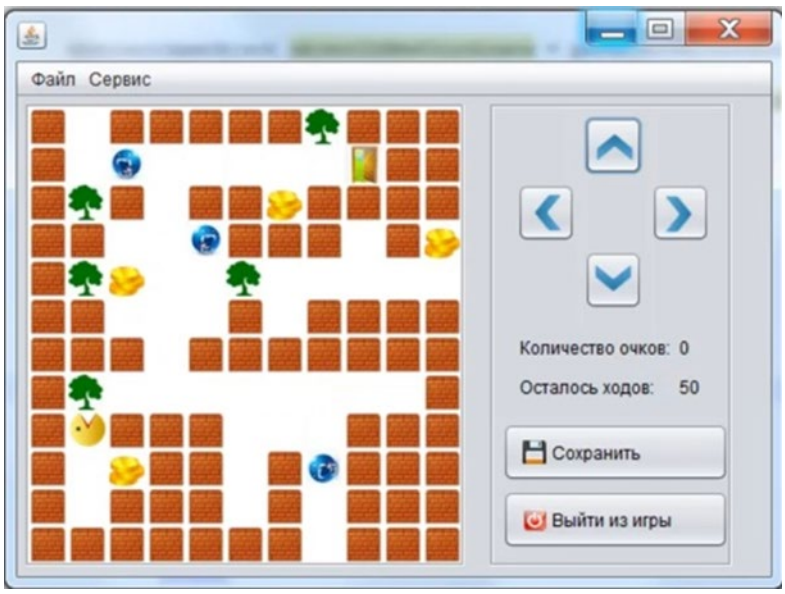
№	Атрибуты качества ПО группы «Usability»	Описание
1	Распознаваемость соответствия	степень узнаваемости, с которой пользователи могут распознать, соответствует ли продукт или система их потребностям
2	Обучаемость	степень, в которой продукт или система могут использоваться определенными пользователями для достижения определенных целей обучения использованию продукта или системы с эффективностью, результативностью, свободой от риска и удовлетворением в определенном контексте использования
3	Используемость (операбельность)	степень работоспособности, при которой продукт или система обладают свойствами, облегчающими их эксплуатацию и контроль
4	Защита от ошибок пользователя	степень защиты пользователя от ошибок, в которой система защищает пользователей от совершения ошибок
5	Эстетика GUI	степень, в которой пользовательский интерфейс обеспечивает приятное взаимодействие для пользователя
6	Доступность	степень, в которой продукт или система могут использоваться людьми с широчайшим спектром характеристик и возможностей для достижения определенной цели в определенном контексте использования

Оценка атрибутов качества ПО группы «Usability»:

1. Распознаваемость соответствия.

В игре имеется просто удобное меню, в котором используется русский язык, меню интуитивное, понятное, название кнопок отражает суть выполняемой ими функции.





2. Обучаемость.

После нескольких попыток прохождения игры, игрок обучиться проходить ее максимально быстро, поймет логику игры и основные функции объектов. Даже при смене игровой карты, в которой меняется положение объектов и стратегии поведения монстров, игрок быстро сориентируется и поймет суть игры. Также отдельно на экране отображается остаток ходов и количество очков, которые игрок получает при сборе сокровищ.

3. Используемость (операбельность)

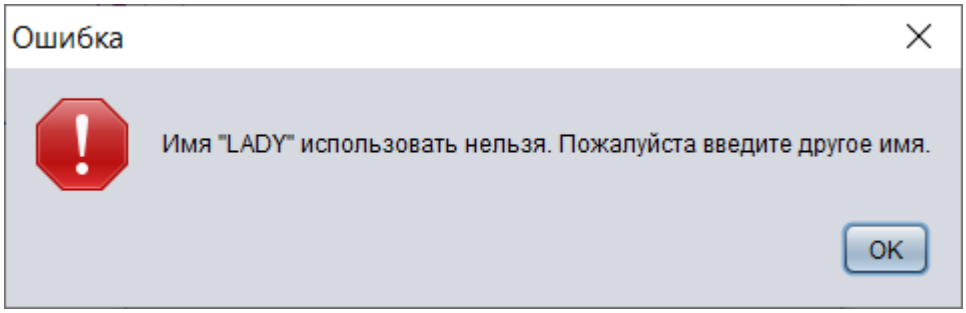
Приложение полностью соответствует ожиданиям пользователя и имеет простой функционал.

4. Защита от ошибок пользователей.

Действия, в случае воздействия различных условий:

- приложение не зависает и не тормозит при быстром переключении кнопок управления;
- приложение хранит данные пользователя в БД, поэтому они не потеряются;
- если система по разным причинам перестает работать (что-то с хост-машиной, либо сбой в системе), то игра сохраняется автоматически.

В случае, если пользователь ввел запрещенное имя, на экране отобразиться следующее сообщение об ошибке.



5. Эстетика GUI

Пользовательский интерфейс обеспечивает приятное взаимодействие для пользователя, используется приятное графическое отображение, спокойные цветовые тона, также каждое действие игрока на карте имеет приятное звуковое сопровождение, которое отражает суть происходящего процесса. При различных движениях пользователя звук меняется: движение, скрытие за деревом, собирание золота, проигрыш, выход.

6. Доступность.

Игра является однопользовательской и не предусматривает режим с несколькими игроками. Игра не требует специальных технических навыков от пользователей или наличия высшего образования. Возрастной диапазон не ограничен, это может быть любой пользователь с базовым уровнем владения компьютером.

Для оценки юзабилити ПО было проведено юзабилити-тестирование – это способ тестирование, при котором определяется удобство, понятность и привлекательность продукта для пользователя. Выполняется с помощью привлечения пользователей с целью прохождения тестов, с целью получении в дальнейшем информации и выводов от тестировщиков.

Участникам юзабилити-тестирования было необходимо выполнить следующие задачи:

1) начать игру;

2) сохранить игру;

3) восстановить игру;

4) посмотреть статистику игр;

5) собрать все сокровища;

6) умереть от монстра;

7) пройти игру в лимитированное количество ходов.

Все задачи были выполнены пользователями успешно, без затруднений и проблем. Респонденты отметили простоту управления и сложность игры. Юзабилити-тестирование было пройдено успешно, поэтому как таковых замечаний по улучшению UX не было.